VŠB - Technická univerzita Ostrava

Fakulta strojní

Katedra automatizační techniky a řízení

VÝVOJ APLIKACE PRO OVLÁDÁNÍ ROBOTICKÉ KOULE SPHERO V OPERAČNÍCH SYSTÉMECH WINDOWS

SPHERO APPLICATION DEVELOPMENT IN OPERATING SYSTEMS WINDOWS

Student: Robert Stavěla

Vedoucí práce: doc. Ing. Marek Babiuch, Ph.D.

Ostrava 2016

VŠB - Technická univerzita Ostrava
Fakulta strojní
Katedra automatizační techniky a řízení

Zadání bakalářské práce

Student: **Robert Stavěla**

Studijní program: B2341 Strojírenství

Studijní obor: 3902R001 Aplikovaná informatika a řízení

Téma: Vývoj aplikace pro ovládání robotické koule Sphero v operačních systémech Windows
Sphero Application Development in Operating Systems Windows

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Seznamte se s parametry robotické koule Sphero a dostupnými aplikacemi, které stručně popište.
2. Analyzujte vývojové možnosti aplikací pro Sphero v systému Windows.
3. Prostudujte princip vývoje Windows aplikací ve vhodné verzi vývojového studia VS .NET.
4. Implementujte aplikaci ovládání funkcí robotické koule v systému Windows.
5. Popište implementované funkce, zdokumentujte výslednou aplikaci a zhodnoťte dosažené výsledky.

Seznam doporučené odborné literatury:

ALBAHARI, B., DRAYTON, P., MERRILL, B. C# Essentials., O'Reilly Media, 2002, 216 s., ISBN 0-596-00315-3.
HILYARD, J., TEILHET, S. C# 3.0 Cookbook., O'Reilly Media, 2007. 886 s., ISBN 0-596-51610-X.
KAČMÁŘ, D. Programujeme .NET aplikace ve Visual studiu .NET., Praha: 2001, 335 s. ISBN 80-7226-569-5.
PROSISE, J. Programování v Microsoft .NET., Computer Press, 2003, 736 s., ISBN 80 7226 879 1.
Sphero Developer center <online> available at: https://developer.gosphero.com/

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Marek Babiuch, Ph.D.**

Datum zadání: 11.12.2015

Datum odevzdání: 16.05.2016

doc. Ing. Renata Wagnerová, Ph.D.
vedoucí katedry

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta strojní 3 OSTRAVA

doc. Ing. Ivo Hlavatý, Ph.D.
děkan fakulty

Místopřísežné prohlášení studenta

Prohlašuji, že jsem celou bakalářskou práci včetně příloh vypracoval samostatně pod vedením vedoucího bakalářské práce a uvedl jsem všechny použité podklady a literaturu.

V Ostravě dne 16. 5. 2016 .....

podpis studenta

Prohlašuji, že

- jsem byl seznámen s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo.
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen „VŠB-TUO“) má právo nevýdělečně ke své vnitřní potřebě bakalářskou práci užít (§ 35 odst. 3).
- souhlasím s tím, že diplomová práce bude v elektronické podobě uložena v Ústřední knihovně VŠB-TUO k nahlédnutí a jeden výtisk bude uložen u vedoucího diplomové práce. Souhlasím s tím, že údaje o kvalifikační práci budou zveřejněny v informačním systému VŠB-TUO.
- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona.
- bylo sjednáno, že užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).
- beru na vědomí, že odevzdáním své práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, bez ohledu na výsledek její obhajoby.

V Ostravě dne 16. 5. 2016 .....

Podpis

Robert Stavěla

Obora 668

75701 Valašské Meziříčí

Poděkování

Chtěl bych poděkovat svému vedoucímu doc. Ing. Marku Babiuchovi, Ph.D. za vedení a pomoc při vypracovávání mé bakalářské práce a doc. Ing. Renatě Wagnerové, Ph.D. za rady a věcné připomínky.

Anotace

STAVĚLA, R. *Vývoj aplikace pro ovládání robotické koule Sphero v operačních systémech Windows: Bakalářská práce.* Ostrava: Vysoká škola báňská – Technická univerzita Ostrava, 2016, Vedoucí práce: doc. Ing. Marek Babiuch, Ph.D.

Tato bakalářská práce se zabývá vývojem aplikace na ovládání robotické koule Sphero, která je vytvořena pro operační systémy Windows. K vývoji je využito prostředí Visual Studio společnosti Microsoft. V práci jsou popsány části SDK knihoven vydané tvůrci robotické koule, bez kterých by nebylo možné aplikaci vyvíjet. Nově je zde vytvořeno nové grafické prostředí a implementovány nové funkce. Všechny vytvořené části jsou podrobně popsány v dokumentu. Další částí této práce jsou návody do cvičení, podle kterých si budou studenti moci osvojit základy značkovacího jazyku XAML.

Klíčová slova:

Sphero, robotická koule, aplikace, vývoj, Windows, Visual Studio

Annotation

STAVĚLA, R. *Sphero Application Development in Operating Systems Windows: Bachelor thesis.* Ostrava: Vysoká škola báňská – Technical University of Ostrava, 2016, Thesis head: doc. Ing. Marek Babiuch, Ph.D.

This bachelor's thesis deals with application development to control the robotic ball Sphero. This application is designed for Windows operating systems and for development is used Microsoft Corporation's Visual Studio environment. The work describes parts of SDK libraries released by robotic ball creators. New graphic interface and implemented new function are made in this application. All these created parts are described in detail in the document. Next part of this work are instructions for exercises, which enable to students learn the basics of XAML mark-up language.

Key words:

Sphero, robotic ball, application, development, Windows, Visual Studio

Obsah

Seznam použitých symbolů a zkratek

API	Programovací rozhraní (*Application Programming Interface)*
g	Jednotka hmotnosti *(Gram)*
h	Jednotka času *(Hodina)*
IDE	Vývojové prostředí *(Integrated Development Environment)*
iOS	Operační systém
m	Jednotka vzdálenosti *(Metr)*
m/s	Jednotka rychlosti *(Metr za sekundu)*
mAh	Jednotka náboje, kapacita baterie *(Miliampérhodina)*
PNG	Grafický formát *(Portable Network Graphics)*
RGB	Barevný model *(Red Green Blue)*
SDK	Sada vývojových nástrojů *(Software development kit)*
V	Jednotka elektrického napětí *(Volt)*
WPF	Technologie tvorby aplikací *(Windows Presentation Foundation)*
XAML	Značkovací jazyk *(Extensible Application Markup Language)*

1 Úvod

Aplikace bude určena zejména pro mobilní zařízení jako chytrý telefon (smartphone) či tablet. Jelikož operační systémy Android nebo iOS mají rozsáhlejší databázi softwarů, bude tato aplikace vyvíjena pro operační systém Windows. Díky otevřenosti firmy Orbotix jsou volně dostupné knihovny (SDK), potřebné pro vytvoření aplikace. Obsahují veškeré instrukce pro robotickou kouli, jako jsou příkazy ke směru jízdy, změny barev, nastavení referenčního bodu jízdy a další. Tyto sady nástrojů jsou navíc volně šiřitelné.

Před samotnou prací nad vývojem aplikace, bude nutné seznámit se s robotickou koulí, s jejími parametry, funkčností a možnostmi jak s ní pracovat. Pochopit jak pracuje její hardware, z čeho se skládá a jak funguje. Neméně důležité bude prozkoumat již dostupné aplikace, které ukážou, jak se dá Sphero využít a jaké nabízí možnosti. Protože aplikace bude sloužit k zařízením s operačním systémem Windows, budu se zabývat možnostmi programování v tomto systému. Nejvhodnější možností vývoje aplikace, která bude pracovat na platformě Windows je vývojové prostředí Microsoft Visual Studio a to díky širokému spektru možností, které nabízí.

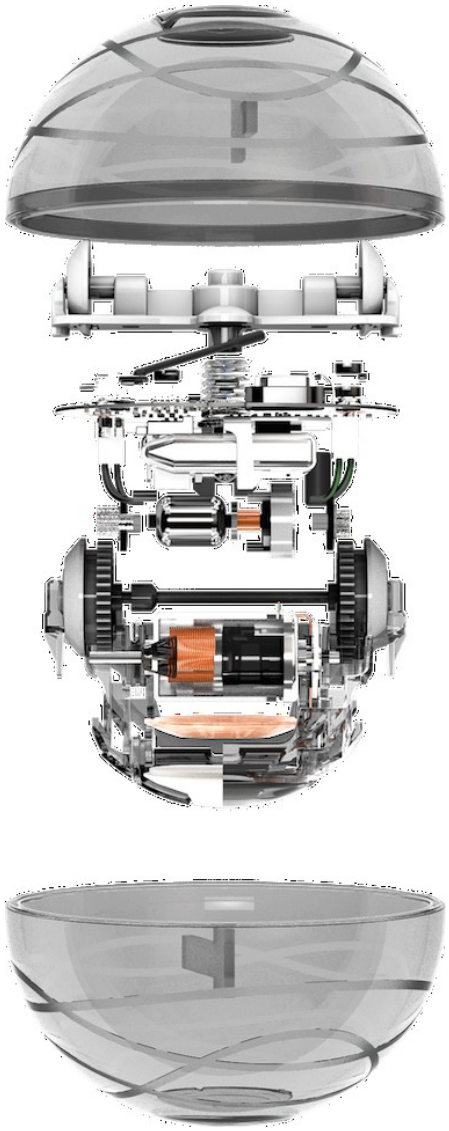
Toto téma jsem si vybral, jelikož mě zaujalo propojení strojírenství s informatikou a je to bezpochyby perspektivním spojením do budoucna. Také mě zajímá, jak pracují různé aplikace či programy, jejich vývoj a co se skrývá uvnitř těchto softwarů. Lákalo mě „oživování“ strojů či hardwarů a jejich automatizace.

2 Sphero koule

Sphero je ovladatelná robotická koule, jejíž pohyb a směr pohybu lze kontrolovat prostřednictvím chytrého telefonu nebo tabletu. Přístroj, pomocí kterého je koule ovládána, musí být vybaven příslušnou aplikací, prostřednictví které se spojí se Sphero koulí. Komunikace mezi oběma zařízeními je realizována díky bezdrátovému spojení Bluetooth s dosahem až 20 metrů. Zdroj elektrické energie tvoří baterie s kapacitou 350mAh. Ta má výdrž přibližně jednu hodinu aktivního používání a nabíjení trvá tři hodiny. Nabíjecí adaptér má podobu žlábku, do kterého koule přesně padne a nabíjí se bezdrátově indukcí. Nabíjecí dok je napájen kabelem s koncovkou do zásuvky na 230V. Konstrukci tvoří dvě duté polokoule pevně slepené dohromady a materiál je vysoce odolný tvrzený polykarbonát. Koule má průměr 74mm (velikost přibližně stejná jako tenisový míček) a váží 168g. Navíc díky dvěma RGB diodám dokáže vytvořit šestnáct miliónů odstínů barev. Maximální rychlost koule je 1m/s. [Mysliveček, 2015]

Tab. 1 – Technické parametry [Mysliveček, 2015]

Dosah	20m
Kapacita baterie	350mAh
Doba nabíjení	3h
Napájení	230V
Váha	168g
Rozměry	Ø74mm
Maximální rychlost	1m/s



Obrázek 1 - Pohled do nitra Sphero koule [Sphero.com, 2015]

2.1 Vznik

Sphero vzniklo jako ukázkový předmět nové technologie, kterou vytvořila skupinka amerických inženýrů. Technologie byla zprvu určena pro strojírenský průmysl, ale ukázka v podobě robotické koule se uchytila jako hrací předmět pro široké spektrum uživatelů. Tvůrci robotické koule také uvolnili knihovny volně ke stažení pro ty, kteří si chtějí svou aplikaci vytvořit a sdílet ji.

2.2 Kompatibilita

S kompatibilitou by neměl být větší problém. Aplikace pro ovládání Sphera jsou dostupné pro android již u starších verzí. Co se týče iOS systému, tak je spustíme od verze iPhone 3G až po nejnovější. Sphero a jeho ovládací aplikace jsou dostupné na všech typech operačních systémů, výjimkou tedy není ani Windows Phone. Pro Windows používaných na chytrých zařízeních jsou čerstvě vydané knihovny (tzv. API) k volnému stažení. Touto verzí systému se bude také zaobírat tato bakalářská práce a bude na ni vytvořena nová aplikace.

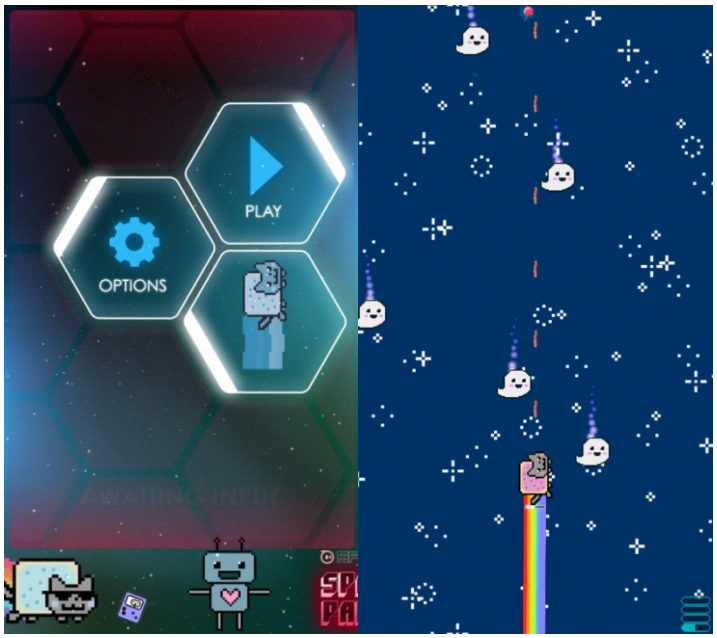
2.3 Dostupné aplikace

Nejzákladnější software je stejnojmenná aplikace *Sphero*. Tato aplikace umožňuje dvě možnosti hraní. Tou první je *„Just drive“* (prostě jízda), prostřednictvím které robotickou kouli pouze ovládáme a jezdíme s ní. Určování směru jízdy a akceleraci je možno korigovat dotykově, pomocí ovládače zobrazeném na displeji mobilu nebo tabletu. Je zde i možnost nastavení rychlosti koule. V každé aplikaci na ovládání Sphera je kalibrace, kterou se manuálně nastaví orientace robotické koule vůči směru ovládání. Máme zde na výběr také triky, například kotoul, tanec, kdy koule „zatancuje“ nastavenou sestavu se světelnými efekty, duha, tento trik při jízdě střídá všechny barvy nebo turbo pro větší zrychlení. Tyhle triky můžeme kupovat za body, které získáváme prostou jízdou. Druhou možností je „Level-up“. V téhle části máme stejné prostředí, ale za získané body se otevírají různé mise například po určitou dobu mít zapnuté při jízdě turbo, třikrát po sobě nabourat a podobně.

Obrázek 2 - Hlavní plocha aplikace *Sphero*

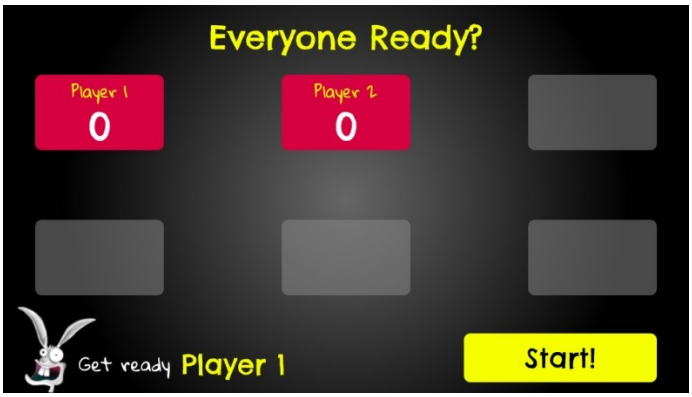
Další aplikací je *Draw&Drive*, kde na plátno dotykem nakreslíme požadovanou dráhu, kterou by mělo Sphero opsat. Barvu, jakou bude koule za jízdy svítit, měníme nastavením barvy čáry v aplikaci.

S aplikací Nyan Cat SpaceParty! nebo zkráceně *SpaceParty!* si můžeme zahrát i bez Sphero koule a to jen za pomoci G-senzoru v mobilním zařízení. Můžeme ale také hru ovládat robotickou koulí, a nakláněním dopředu, dozadu a do boků, ovládat létajícího hrdinu v aplikaci. Otáčením kolem osy pak ovlivňujeme směr střelby na protivníky. Cílem hry, je přes nálety soupeřů doletět vesmírem až do galaxie, která představuje konec kola.



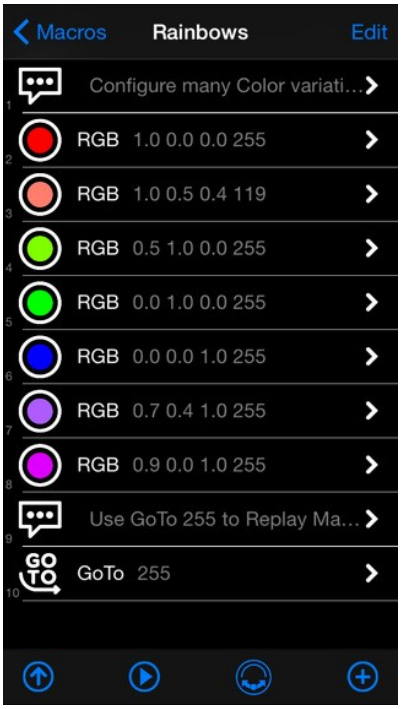
Obrázek 3 - Prostředí hry *SpaceParty!*

ColorGrab nám umožňuje hru více hráčů. Aplikace v každém kole automaticky vylosuje barvu a úkolem hráče, který je zrovna na řadě, je chytit a zastavit kouli, která se točí a mění barvy, zrovna na té barvě, co nám aplikace vylosovala. Čím dřív to hráč zvládne, tím víc dostane bodů. Vyhrává hráč, který jako první dosáhne předem nastavených bodů.



Obrázek 4 - Skóre hráčů ve hře *ColorGrab*

Aplikace *MacroLab* umožňuje naprogramovat sestavu pohybů Sphero koule, pomocí příkazů, které jsou k dispozici. Ty pak sestavujeme postupně za sebou v tom pořadí, ve kterém by měly být vykonány. Kromě pohybových příkazů jako kutálení, orientace, či rotace zde můžeme nastavit i barvu. Ta může klasicky svítit nebo se může nastavit i prolínání mezi více barvami. V aplikaci jsou také již nadefinované sestavy jako udělat okruh dráhy ve tvaru osmičky a podobně.



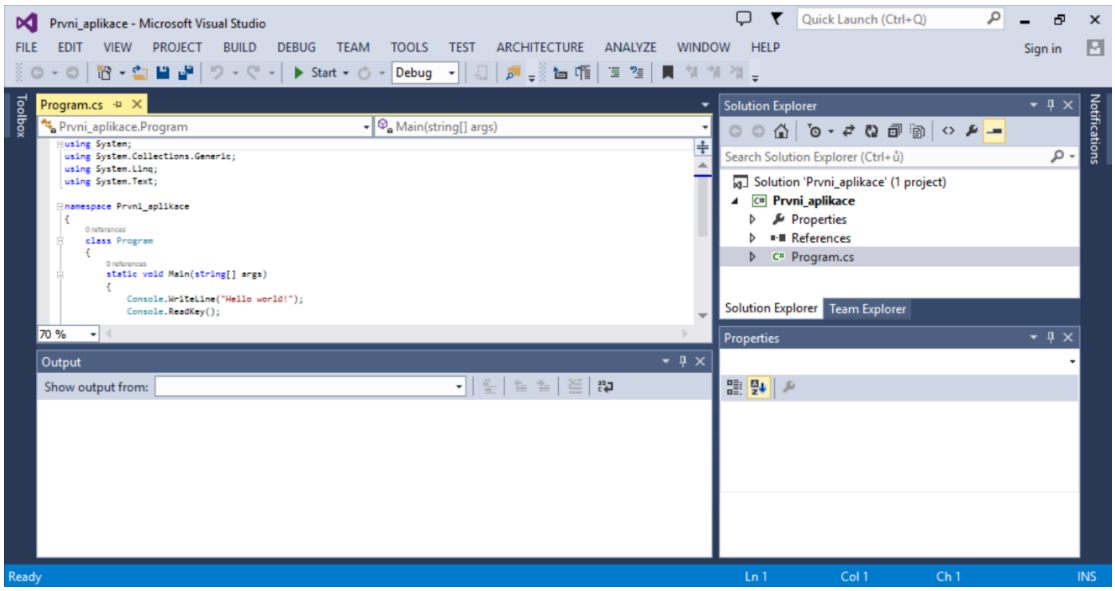
Obrázek 5 - Vzhled aplikace *MacroLab* [Itunes.apple.com, 2016]

3 Možnosti vývoje aplikací pro Sphero

Nejjednodušší a zároveň nejzákladnější programování Sphero koule umožňuje dříve popsaná aplikace *MacroLab*. Tou lze naprogramovat ale jen pohyby a sestavy pohybů jak bylo popsáno v předcházející kapitole. Pro vytváření aplikací s vlastním prostředím, grafickým zpracováním, cílem či funkcí se využívají různá vývojová prostředí, tzv. IDE, což jsou software usnadňující programování za pomoci různých programovacích jazyků (např.: Object Pascal, JAVA nebo C a C++). Vývojových prostředí je celá řada. V této bakalářské práci se však bude využívat balíček nástrojů firmy Microsoft, Visual Studio 2013. [Wikipedia, 2015]

3.1 Microsoft Visual Studio

Visual Studio je vývojové prostředí pro vytváření aplikací s grafickým rozhraním, webových aplikací, webových stránek nebo webových služeb na různých platformách. U firmy Microsoft jsou to: Microsoft Windows, Windows Mobile, .NET nebo Microsoft Silverlight. VS podporuje také vytváření aplikací pro Android či iOS. Visual Studio podporuje více programovacích jazyků už ve výchozím nastavení (C, C++, VB.NET, C#).



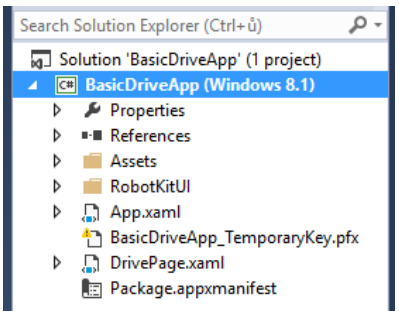
Obrázek 6 - Prostředí programu Visual Studio

Podpora je zajištěna pomocí balíčků Visual C++, Visual Basic .NET a Visual C#. Rozmach podpory pro Visual Studio je obrovský jedná se o nástroje buď přímo od

Microsoftu nebo o externí nástroje a služby. Nástrojové balíčky a služby lze jednoduše nainstalovat při instalaci programu a mohou být doinstalovány i zvlášť z rozsáhlé knihovny. Doplňky mohou být placené i bezplatné. Visual Studio nabízí velké množství výhod, které dokáží velice usnadnit práci. Jedná se hlavně o integrované možnosti programování a opakovatelnost kódu, například při sdílení projektů (jedna aplikace pro Windows, Android a iOS) můžeme sdílet napsaný kód na více platformách. Další výhodou je také například automatické doplňování kódu, které zvyšuje přesnost a rychlost psaní programu. Při chybách psaného kódu nebo při špatném formátování strukturovaného textu, ukazuje Visual Studio chyby a upozornění a zároveň nabízí automatické opravy. [Microsoft, 2016]

3.2 Knihovny SDK

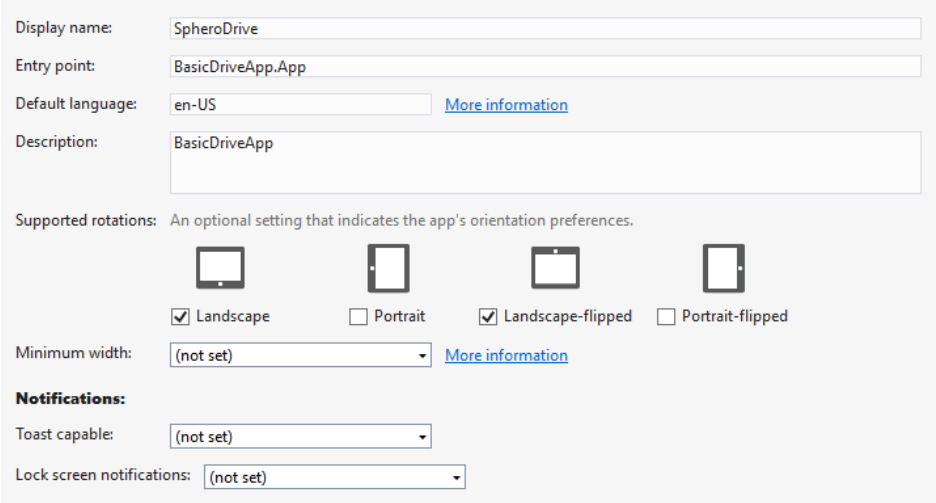
Software development kit je balíček nástrojů pro vývoj softwaru na různých platformách. Tento balíček je určen pro programátory či vývojáře, kteří chtějí vyvíjet aplikaci pro různé mobilní a počítačové systémy, platformy herní konzole a další. Společnost, která vytvořila Sphero vydala tyto volně šiřitelné knihovny pro více platforem (iOS, Android, Windows 8.1) a na oficiálních stránkách jsou k dispozici také neoficiální knihovny (Ruby, Python, Mac).



Obrázek 7 - Obsah SDK knihovny

Po stažení balíčku pro Windows je nutné jej otevřít ve vývojovém prostředí Visual Studio. Balíček obsahuje základní funkce a příkazy pro Sphero, základní grafické rozvržení, nastavení základní aplikace a další. Obsah balíčku je formou stromu souborů, jehož kapitoly jdou dále rozbalovat a otvírat.

V souboru *Package.appxmanifest* můžeme nalézt různá nastavení jako je jméno aplikace, popis aplikace, jazyk nebo podporované natočení obrazovky při spuštění aplikace.



Obrázek 8 - Část obsahu souboru *Package.appxmanifest*

Po rozbalení souboru *DrivePage.xaml* je na výběr další soubor, ve kterém se nachází funkční část kódu značkovacího jazyka XAML a výchozí nastavení chování Sphera za určitých podmínek nebo situací, jako jsou instrukce po připojení Sphera, konfigurace joysticku nebo hledání dostupných zařízení Sphero. V obrázku je vidět příkaz pro hledání Sphera, ke kterému by se dalo připojit.

```
SpheroName.Text = kNoSpheroConnected;

RobotProvider provider = RobotProvider.GetSharedProvider();
provider.DiscoveredRobotEvent += OnRobotDiscovered;
provider.NoRobotsEvent += OnNoRobotsEvent;
provider.ConnectedRobotEvent += OnRobotConnected;
provider.FindRobots();
```

Obrázek 9 - Ukázka kódu - hledání Sphera

Dále je zde také konfigurace akcí pro ovládání, které se provedou po doteku na ovládací joystick, zajišťující směr, kterým Sphero koule pojede, paletu barev, pro nastavení aktuální barvy a kalibrační prvek pro nastavení referenčního bodu, od kterého se dále odvíjí jízda.

```
m_colorwheel = new ColorWheel(ColorPuck, m_robot);
m_joystick = new Joystick(Puck, m_robot);

m_calibrateElement = new CalibrateElement(
    CalibrateRotationRoot,
    CalibrateTarget,
    CalibrateRingOuter,
    CalibrateRingMiddle,
    CalibrateRingInner,
    CalibrationFingerPoint,
    m_robot);
```

Obrázek 10 - Ukázka kódu - konfigurace ovládání

Velice důležité je popsání pro vyhledání a připojení robotické koule. Zde je vidět, že když je nalezeno nějaké zařízení Sphero, objeví se jeho nastavené jméno a poté se začne připojovat. Jestliže je správně připojeno, je také v aplikaci vidět jméno připojeného Sphera. Tento popis v programu dále pokračuje tak, že když nebylo nalezeno žádné příslušné zařízení, objeví se dialog se zprávou o nenalezení Sphera.

```
Debug.WriteLine(string.Format("Discovered \"{0}\"", robot.BluetoothName));

if (m_robot == null)
{

    RobotProvider provider = RobotProvider.GetSharedProvider();
    provider.ConnectRobot(robot);
    ConnectionToggle.OnContent = "Připojování...";
    m_robot = (Sphero)robot;
    SpheroName.Text = string.Format(kConnectingToSphero, robot.BluetoothName);
}
```

Obrázek 11 - Ukázka kódu - připojení Sphera

Jestliže je Sphero připojeno, můžeme nastavovat barvu, kterou bude koule svítit. Základem jsou dvě RGB diody, které mohou vytvořit přibližně 16 miliónů barev. To znamená, že každá barva (červená, zelená a modrá) lze nastavit 255 odstíny a ty se mezi sebou kombinují.

```
Debug.WriteLine(string.Format("Connected to {0}", robot));
ConnectionToggle.IsOn = true;
ConnectionToggle.OnContent = "Připojeno";
m_robot.SetRGBLED(255, 255, 255);

SpheroName.Text = string.Format(kSpheroConnected, robot.BluetoothName);
SetupControls();
```

Obrázek 12 - Ukázka kódu - akce po připojení

Možností je také snímat a zobrazovat data z akcelerometru a gyroskopu. Tyto informace jsou snímány ve třech jednotlivých osách (X, Y, Z).

```
AccelerometerX.Text = "" + reading.X;
AccelerometerY.Text = "" + reading.Y;
AccelerometerZ.Text = "" + reading.Z;
```

Obrázek 13 - Ukázka kódu - data akcelerometru

V Obsahu se také nachází složka *RobotKitUI*, ve které je mimo jiné také soubor *CalibrateElement.cs*. Ten popisuje nastavení kalibračního bodu, od kterého se odvíjí směr jízdy. V této části kódu (obr. 14) je popsáno nastavení pozice tohoto bodu na Sphero kouli v závislosti na pohybu bodu v aplikaci. Předchází jí část kódu, která popisuje, že po stisknutí a držení tlačítka na kalibraci se rozsvítí na Sphero kouli dioda ukazující tento kalibrační bod a následuje ji také část popisující puštění tohoto bodu v aplikaci, která zapříčiní zhasnutí diody ukazující kalibrační bod na Spheru.

```
private void OnPointerMoved(object sender, PointerRoutedEventArgs args) {
    m_rotation.Angle = 0;
    Windows.UI.Input.PointerPoint pointer = args.GetCurrentPoint(m_calibrateRotationRoot);
    if (pointer.Properties.IsLeftButtonPressed) {
        CalibrateFromPoint(pointer.Position);
    }
}
```

Obrázek 14 - Ukázka kódu - kalibrační bod

Podobně jako u kódu pro ovládání kalibračního bodu je to také pro ovládání joysticku. Zde je ukázaná část kódu, který monitoruje pozici prstu na joysticku v aplikaci a tyto informace v závislosti na ose X a Y posílá do Sphero koule. Ta je poté vyhodnotí a vykonává podle nich samotný pohyb. Zde rovněž předchází část kódu pro snímání, kdy se přiloží prst na joystick v aplikaci a následuje popsání po puštění prstu z joysticku.

```
private void PointerMoved(object sender, PointerRoutedEventArgs args)
{
    Windows.UI.Input.PointerPoint pointer = args.GetCurrentPoint(null);
    if (pointer.Properties.IsLeftButtonPressed)
    {
        Point newPoint = pointer.RawPosition;
        Point delta = new Point(
            newPoint.X - m_initialPoint.X,
            newPoint.Y - m_initialPoint.Y);
        m_translateTransform.X = delta.X;
        m_translateTransform.Y = delta.Y;
        args.Handled = true;

        ConstrainToParent();

        SendRollCommand();
    }
}
```

Obrázek 15 - Ukázka kódu - nastavení joysticku

3.3 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní v této aplikaci znamená rozhraní, které obsahuje funkční graficky zobrazené ovládací prvky, jako jsou tlačítka, posuvníky, přepínače, textové pole a další. V této aplikaci je využita technologie WPF (Windows Presentation Foundation). Způsob WPF nabízí obsáhlé zdroje již zmíněných ovládacích prvků. Pro grafický návrh bude využit značkovací jazyk XAML, který umožňuje vytvářet grafické rozhraní aplikace s oddělenou funkční části, v tomhle případě je to kód v jazyce C#. [Vávra, 2014]

Jako příklad může pomoci zobrazování dat naměřených tříosým akcelerometrem a gyroskopem. V souboru *DrivePage.xaml* se nachází kód pro vytvoření textových polí s názvem, o který prvek se jedná (akcelerometr a gyroskop) a pod každým z nich jsou další 3 textové pole s jednotlivými osami prvků (x, y, z).

```
    <TextBlock x:Name="AccelerometerValues" HorizontalAlignment="Right"
Text="Accelerometer" VerticalAlignment="Top" FontSize="24"/>

    <TextBlock x:Name="AccelerometerX" HorizontalAlignment="Right" Text="x"
VerticalAlignment="Top" FontSize="14"/>

    <TextBlock x:Name="AccelerometerY" HorizontalAlignment="Right" Text="y"
VerticalAlignment="Top" FontSize="14"/>

    <TextBlock x:Name="AccelerometerZ" HorizontalAlignment="Right" Text="z"
VerticalAlignment="Top" FontSize="14"/>
```

Obrázek 16 - XAML - data akcelerometru

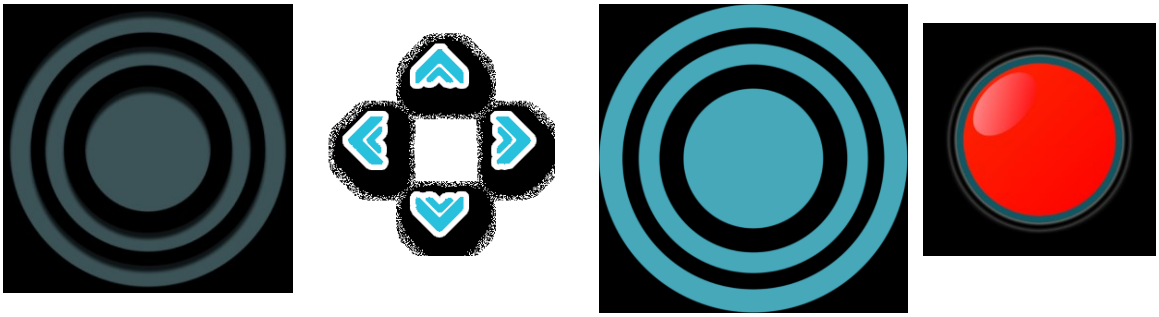
Příkaz *x:Name=“ ... “* slouží k načtení proměnné, která se potom zobrazuje v samotné aplikaci. Tato proměnná (například: “AccelerometerX“) je uložena ve funkční části tohoto grafického rozhraní v souboru *DrivePage.xaml.cs.*

Příkaz (obr. 13) vyvolává data z akcelerometru a čte je jako text. Ten je potom načítán do souboru s příponou *xaml* a převáděn na text viditelný v aplikaci (obr. 17).

Obrázek 17 - Zobrazení dat akcelerometru

4 Tvorba grafického prostředí aplikace

Pro vytvoření grafického prostředí aplikace je nejprve nutné navrhnout, jak by měla vypadat, tudíž vytvořit si skicu s rozvržením jednotlivých prvků v okně aplikace. Mým požadavkem bylo, aby v okně vynikal joystick pro ovládání směru jízdy. V grafickém editoru jsem si vytvořil návrhy různých komponentů, které jsem poté použil (obr. 18). Ovládací joystick se skládá ze dvou částí, a to z pozadí a posuvného ukazatele směru jízdy. Dále jsem si vytvořil tlačítko na kalibraci Sphero koule a obrázek na pozadí. Vytvořené obrázky je třeba uložit ve formátu PNG nejen pro dobrou kvalitu a rozsah barev ale hlavně z důvodu průhlednosti některých míst, které by jinak byly vyplněné jinou barvou.



Obrázek 18 - Prvky grafického prostředí aplikace

Tyto obrázky jsem potom musel vložit jako existující objekt do projektu ve Visual Studiu. Po vložení bylo nutné připojit tento obrázek k příslušnému objektu. Následuje krok pro vytvoření tabulky – *Grid*. Tento příkaz na vytvoření tabulky se využívá mimo jiné k ohraničení objektu tak, že bude obsahovat pouze jeden sloupec a jeden řádek.

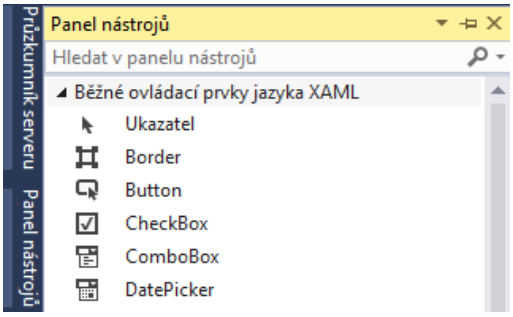
Umístění všech objektů vložených v okně aplikace je relativní, to znamená, že jejich pozice je určena tím, ke kterému okraji je objekt přichycen. K přichycení slouží příkaz *HorizontalAlignment* a *VerticalAlignment*, k němuž se přiřazuje hodnota (Top, Right, Bottom, Left a Center). Potom se pomocí funkce *Margin* určí odskok od jednotlivých hran. Toto nastavení zaručuje, že aplikace zachová své grafické proporce na různých zařízeních s různým rozlišením. Další funkcí je *Grid.Background*, která určuje pozadí tabulky (ohraničení) a v tomhle případě (obr. 19) je na pozadí vložen obrázek pomocí funkce *ImageBrush*, kde se parametrem *ImageSource* vyvolá obrázek z jeho umístění.

Dalšími funkcemi je *Height* a *Width*, určující šířku a výšku objektů. Jednotkou velikosti jsou zde obrázkové body.

```
<Grid HorizontalAlignment="Center" Height="400" Margin="615,235,407,175"
VerticalAlignment="Center" Width="400">
            <Grid.Background>
<ImageBrush Stretch="Fill" ImageSource="Assets/DriveWheel/gfx_driveorb6.png"/>
            </Grid.Background>
      <Image x:Name="Puck" HorizontalAlignment="Center" Height="150"
Margin="83,108,117,92" VerticalAlignment="Center" Width="150"
Source="Assets/DriveWheel/gfx_joystick4.png"/>
</Grid>
```

Obrázek 19 - XAML - Přidání joysticku

Ke vkládání objektů může sloužit také panel nástrojů ve Visual Studiu. Tento panel nástrojů obsahuje veškeré nástroje jako textové pole, tlačítka, přepínače, rámečky, vkládání obrázků, tabulek atd.. Po vložení nástroje do pracovního pole, se automaticky vytvoří základ kódu v příslušném souboru s příponou *.xaml*.



Obrázek 20 - Panel nástrojů ve Visual Studiu

V pravém horním rohu jsou umístěna dvě loga: Logo Vysoké školy báňské a logo Katedry automatizační techniky a řízení. Po pravé straně okna aplikace je umístěn ovládací joystick určující směr jízdy koule Sphero. Joystick se skládá ze dvou částí a to z pozadí a z pohyblivého ukazatele směru, u kterého tvoří hranice pohybu jeho pozadí. Pod joystickem je umístěn červený bod, který slouží ke kalibraci koule. Po stisknutí bodu se na jeho místě objeví kruh s kurzorem a ten tahem nastavíme na požadovanou pozici. Dále je na ploše umístěna paleta pro nastavení barvy Sphero koule. Ta je tvořena také dvěma částmi. Pozadí kruhu je tvořeno samotným spektrem barev a na něm je umístěn kurzor pro určení barvy.

Pozadí je tvořeno funkcí *Grid*, která tvoří ohraničení kolem celého okna zařízení. K této funkci je přiřazeno jméno proměnné *rootGrid* a přičtena hodnota určující barvu pozadí.

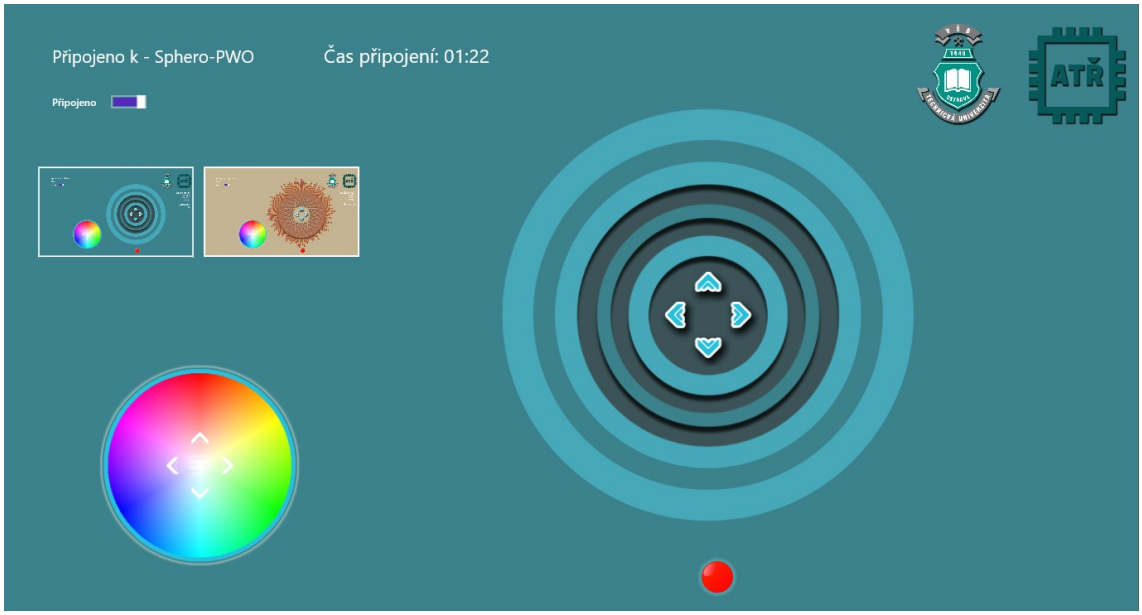
```
<Grid x:Name="rootGrid" Background="Black">
```

Obrázek 21 - XAML - výchozí barva pozadí

Při spuštění aplikace se koule Sphero automaticky připojuje přes Bluetooth se spárovaným zařízením, na kterém je aplikace spuštěna. Jestliže není připojeno žádné zařízení, na pozadí aplikace se automaticky zobrazuje výchozí nastavená barva.

V kódu, který připojuje nalezené zařízení Sphero jsou umístěny příkazy, které se vykonají po připojení. Pro změnu pozadí po připojení je zde nutné vepsat název proměnné *rootGrid.Backround* a k té přiřadit funkci pro vyplnění barvou – *SolidColorBrush*. Argument pro přidání barvy vychází z knihovny, ze které budeme vybírat barvu. Tato knihovna obsahuje výběr asi 50 barev. Je zde ale také možnost nastavení barev klasickou kombinací RGB. V tomhle případě se nastavuje kromě 3 argumentů (červená, zelená a modrá) také argument průhlednost. Nastavení barvy potom má tenhle tvar:

```
rootGrid.Background = new SolidColorBrush(Windows.UI.Color.FromArgb(200, 120, 175,
187));.
```



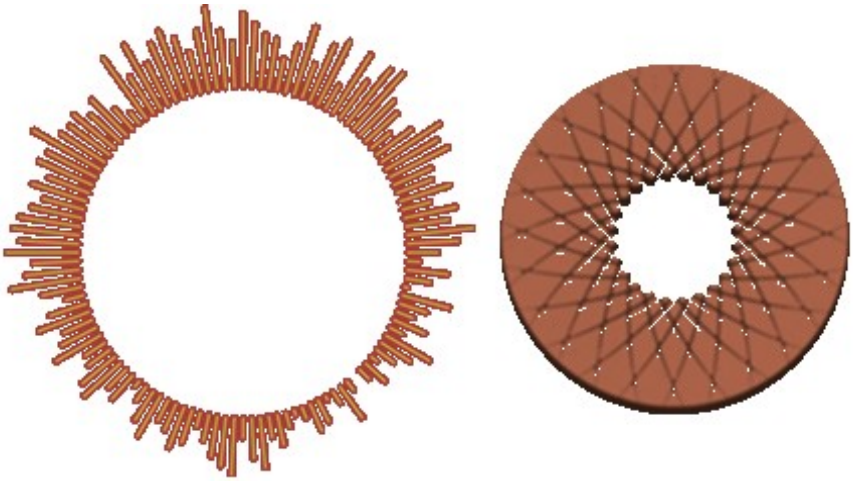
Obrázek 22 - Nastavení barvy pozadí

5 Implementace ovládací funkce

V aplikaci jsou implementovány ovládací funkce, které názorně ukazují, jak lze využít SDK knihovny. U tvorby funkcí je využit značkovací jazyk XAML ve spojení s funkční částí v jazyku C#. Vytvořeny jsou funkce jako změna vzhledu, měření doby připojení ke Spheru a také informační okna s textem.

5.1 Nastavení motivu aplikace

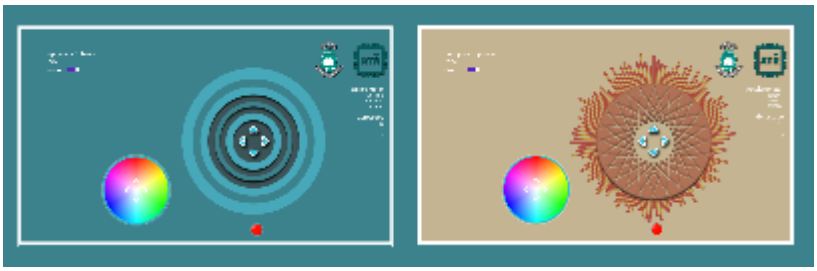
Funkce nastavení motivu v aplikaci bude sloužit ke změně vzhledu aplikace. Změna bude prováděna pomocí tlačítek. Každé tlačítko bude představovat jednu verzi motivu. Tlačítko musí zajišťovat to, že když se zapne první motiv, tak druhý se vypne. Jako první je tedy nutné vytvořit podobu druhého vzhledu. V příslušném softwaru pro grafický návrh musí být vytvořeny prvky dalšího motivu. Tak jako v minulé kapitole jsem si vytvořil obrázek pro pozadí joysticku a obrázek, který joystick zvýrazňuje.



Obrázek 23 - Prvky druhého motivu aplikace

Tyto dva vytvořené obrázky se musí nahrát do projektu jako existující položka, aby s nimi bylo možné pracovat. Obrázky jsou stejně jako u předchozího motivu uloženy ve formátu PNG, pro průhlednost prázdných míst.

Dále musí být vložena tlačítka, která budou představovat jednotlivé verze motivů. Tato tlačítka budou vložena pomocí panelu nástrojů.



Obrázek 24 - Tlačítka na změnu motivu aplikace

Po vložení se vytvoří kódové interpretace tlačítek v souboru *DrivePage.xaml*, tedy v jazyce XAML. Zde je nutné vložit popisek tlačítka, jenž se bude zobrazovat v aplikaci na samotném tlačítku. Vložení popisku zde zajišťuje parametr *Content="..."*. Tyhle dvě tlačítka mají pracovní název „Vzhled1" a „Vzhled2". Dále bude nakonfigurováno jméno proměnné, která zajistí, že po kliknutí nebo klepnutí na tlačítko, vykoná nějakou akci. K zapsání proměnné tady slouží parametr *Click="..."*. Jestliže jsou vytvořena tlačítka stejného typu, je dobré je vložit do rámečku pomocí funkce *Grid*. Celý je nakonec opatřen jménem proměnné *x:Name"..."*.

```
<Grid x:Name="Vzhled" Visibility="Collapsed" HorizontalAlignment="Left"
Height="194" Margin="60,175,0,0" VerticalAlignment="Top" Width="391">
            <Button Content="Vzhled1" HorizontalAlignment="Left"
Margin="54,40,0,0" VerticalAlignment="Top" Click="Button_Click"/>
            <Button Content="Vzhled2" HorizontalAlignment="Left"
Margin="54,113,0,0" VerticalAlignment="Top" Click="Button_Click2"/>
</Grid>
```

Obrázek 25 - XAML - tlačítka pro změnu motivu

Poklepáním na tlačítko v grafickém návrhu se vytvoří proměnná, tedy funkční část kódu v jazyce C#. Tento kód pro ovládání tlačítka se vytvoří v souboru *DrivePage.xaml.cs* a do tohoto kódu budu konfigurovat akce, které se provedou po kliknutí na jednotlivá tlačítka.

```
private void Button_Click(object sender, RoutedEventArgs e)
{

}
```

Obrázek 26 - Ukázka kódu - prázdná proměnná nástroje tlačítko

Dalším krokem je nastavit na prvky, které budou obsahovat jednotlivé styly, viditelnost tak, aby bylo možné dynamicky měnit jejich zobrazení. K tomuto slouží

v jazyce XAML parametr *Visibility="Visible/Collapsed"*, kde se nastavují dva parametry – viditelné, neviditelné. Ke každému prvku bude také přiřazeno jméno proměnné, aby jej bylo možné ovládat.

Zde je vytvořeno pozadí joysticku pomocí rámečku, do kterého je vložen obrázek a ten jej celý vyplňuje. Rámeček obsahuje nastavení jména proměnné, viditelnost, umístění a velikost. Stejným způsobem je vytvořeno druhé pozadí joysticku (druhý motiv), které má však jiné jméno proměnné a je vyplněno jiným obrázkem.

```
<Grid x:Name="PozadiJoy" Visibility="Collapsed" HorizontalAlignment="Center"
Height="400" Margin="0,50,-400,0" VerticalAlignment="Center" Width="400">
      <Grid.Background>
<ImageBrush Stretch="Fill" ImageSource="Assets/DriveWheel/gfx_driveorb6.png"/>
      </Grid.Background>
<Grid>
```

Obrázek 27 - XAML - ovládací joystick

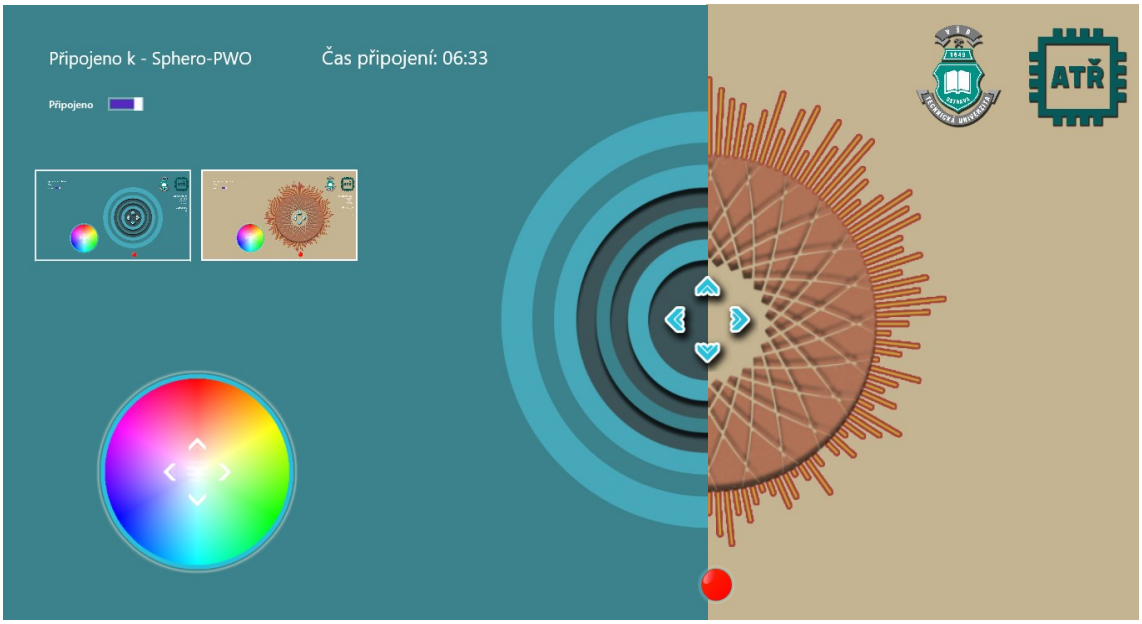
Nyní je třeba nastavit akce, které budou provedeny po kliknutí na jednotlivá tlačítka. To se bude provádět v souboru *DrivePage.xaml.cs* pomocí programovacího jazyka C#. Zde je již vytvořený základní kód tlačítka, který adresuje jméno proměnné akce na vloženém nástroji, u tlačítka to tedy znamená akce po kliknutí. Argument u prvního tlačítka bude odkrývat prvky prvního vzhledu a skrývat prvky dalšího vzhledu. Dále bude u každého motivu nastaveno jiné pozadí. Další tlačítko bude mít opačné argumenty, tedy odkrýt prvky druhého vzhledu a skrýt prvky prvního. Tyto argumenty budou vykonávány pomocí nastaveného parametru *Visibility*, který je nastaven u jednotlivých prvků jednotlivých vzhledů.

```
private void Button_Click(object sender, RoutedEventArgs e)
{

    KoloPrvni.Visibility = Windows.UI.Xaml.Visibility.Visible;
    KoloDruhe.Visibility = Windows.UI.Xaml.Visibility.Collapsed;
    PozadiJoy.Visibility = Windows.UI.Xaml.Visibility.Visible;
    PozadiJoy2.Visibility = Windows.UI.Xaml.Visibility.Collapsed;
    Plocha1 = new SolidColorBrush(Windows.UI.Color.FromArgb(255, 60, 130, 140));
    rootGrid.Background = Plocha1;
    m_robot.SetRGBLED(119, 175, 187);
```

Obrázek 28 - Ukázka kódu - tlačítko pro změnu motivu

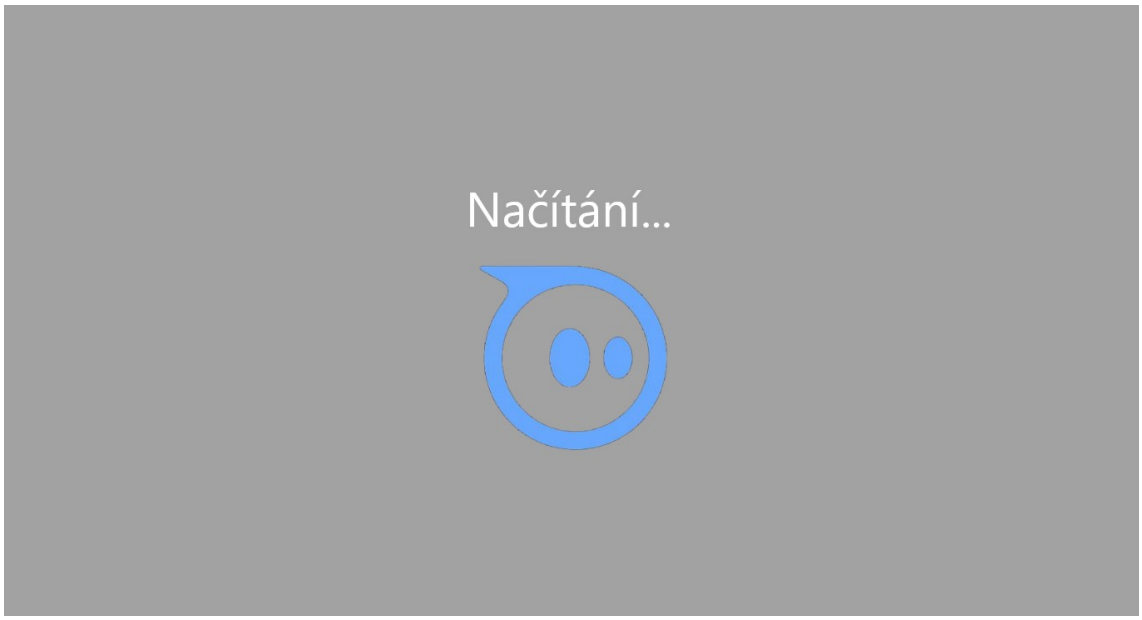
Jednotlivé motivy budou měnit prvky, které jsou k tomu nakonfigurovány a ostatní prvky zůstanou na svém místě nezměněny jako například logo školy, logo katedry a nástroj pro změnu barev Sphero koule.



Obrázek 29 - Prolnutí dvou motivů aplikace

5.2 Vložení načítací obrazovky

Načítací obrazovka je vytvořena pro názorné oddělení prostředí aplikace od části, kdy se čeká na připojení Sphera. Požadavkem na načítací obrazovku je jednoduchost a ať je jasné co za operaci se děje. Na načítací obrazovce je nastaveno neutrální pozadí v podobě šedé barvy. Uprostřed obrazovky je umístěn obrázek se znakem Sphera a nad ním je nápis „Načítání…“.



Obrázek 30 - Vzhled načítací obrazovky

Prvky načítací obrazovky jsou ohraničeny funkcí *grid*. Na tomto ohraničení je definována výchozí viditelnost na hodnotě *„Visible“* tedy viditelné. Ostatní prvky, ať už prvního a druhého motivu nebo stálé prvky grafického prostředí aplikace jsou do připojení Sphera neviditelné.

```
<Grid x:Name="logo1" Visibility="Visible" Height="800" Width="800"
      HorizontalAlignment="Center" VerticalAlignment="Center">
<Image Source="Assets/logo1.png"/>
<TextBlock HorizontalAlignment="Center" Text="Načítání..."
      VerticalAlignment="Bottom" FontStyle="Normal" FontSize="70"/>
</Grid>
```

Obrázek 31 - XAML - Prvky načítací obrazovky

Ve funkční části kódu jsou vytvořeny instrukce, které následují po připojení ke Spheru. K tomu, aby načítací obrazovka fungovala tak jak má, to znamená, že se zobrazí při spuštění aplikace a po připojení Sphera ji nahradí výchozí motiv aplikace, musely být tyto instrukce doplněny ovládacími prvky parametru *Visibility*. Ty jsou použity v návrhu grafického prostředí (XAML). Po připojení ke Spheru tedy musel být parametr *Visibility* u prvků načítací obrazovky změněn na hodnotu *„Collapsed“* – neviditelný. Dále zde musely být přidány kódy pro změnu tohoto parametru i pro prvky výchozího motivu. U těchto prvků by měla být po připojení načtena hodnota *„Visible“* – viditelný. Poté ještě nastavení barvy pozadí, která odpovídá barvě pozadí prvního motivu a také přidání barvy Sphera.

```
private void OnRobotConnected(object sender, Robot robot)
{
        Debug.WriteLine(string.Format("Connected to {0}", robot));
        ConnectionToggle.IsOn = true;
        ConnectionToggle.OnContent = "Připojeno";
        Plocha1 = new SolidColorBrush(Windows.UI.Color.FromArgb(255, 60, 130, 140));
        rootGrid.Background = Plocha1;
        ObrATR.Visibility = Windows.UI.Xaml.Visibility.Visible;
        ObrVSB.Visibility = Windows.UI.Xaml.Visibility.Visible;
        Barva.Visibility = Windows.UI.Xaml.Visibility.Visible;
        Pripojovaci.Visibility = Windows.UI.Xaml.Visibility.Visible;
        Calibrace.Visibility = Windows.UI.Xaml.Visibility.Visible;
        Timer.Visibility = Windows.UI.Xaml.Visibility.Visible;
        Vzhled.Visibility = Windows.UI.Xaml.Visibility.Visible;
        PozadiJoy.Visibility = Windows.UI.Xaml.Visibility.Visible;
        AG.Visibility = Windows.UI.Xaml.Visibility.Collapsed;
        logo1.Visibility = Windows.UI.Xaml.Visibility.Collapsed;
        KoloPrvni.Visibility = Windows.UI.Xaml.Visibility.Visible;
        VnitrekJoy.Visibility = Windows.UI.Xaml.Visibility.Visible;
        MyTimer.Start();

        m_robot.SetRGBLED(119, 175, 187);
```

Obrázek 32 - Ukázka kódu - nastavení viditelnosti prvků

5.3 Časovač připojení

V aplikaci je vytvořen časovač, který měří čas délky připojení Sphera k aplikaci. Časovač je nastaven tak, že po připojení Sphero koule se začne odpočítávat čas. Po odpojení pomocí přepínače se čas opět zastaví.

Nejprve je nutné vytvořit proměnnou *„MyTimer“*, kde bude nakonfigurována třída *DispatcherTimer*, díky které bude možné zpracovávat data z měření času. Dále se musí nastavit interval, jakým se bude čas přičítat a hodnotu intervalu, tedy sekundu.

```
public MainPage()
{
    this.InitializeComponent();
    SetText(new TimeSpan(0, 0, 0));

    MyTimer = new DispatcherTimer();
    MyTimer.Tick += MyTimer_Tick;
    MyTimer.Interval = new TimeSpan(0, 0, 1);
}

1 reference
void MyTimer_Tick(object sender, object e)
{
    time += 1;
    var timeSpan = TimeSpan.FromSeconds(time);
    SetText(timeSpan);
}
```

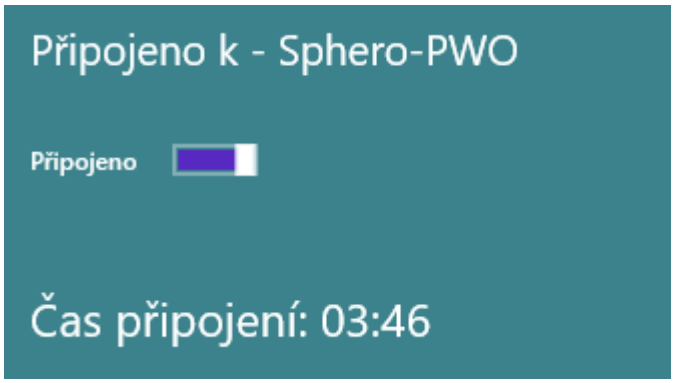
Obrázek 33 - Ukázka kódu - konfigurace časovače

Dále se musí nastavit formát, jakým se bude čas v aplikaci zobrazovat. Hodnota časovače se tedy musí převést na text s čísly a také s textem, který se zobrazuje před časovou hodnotou. Výchozí hodnota je nastavena na „0:00“.

```
private void SetText(TimeSpan timeSpan)
{
    tbTime.Text = String.Format("Čas připojení: {0:00}:{1:00}", timeSpan.Minutes, timeSpan.Seconds);
}
```

Obrázek 34 - Ukázka kódu - nastavení formátu zobrazeného času

Start měření času časovače, který se spustí po připojení Sphera k aplikaci je vytvořen přidáním příkazu pro startování odpočítávání *„MyTimer.Start()“* do odstavce s instrukcemi pro akce po připojení (obr. 32).



Obrázek 35 - Časovač měřící dobu připojení Sphera

5.4 Informační okno

Další funkce je vytvořena pomocí tlačítek, které po kliknutí zobrazí zprávu, v níž je text podávající informace. Zde je první tlačítko *„O Spheru“*, kde je krátce popsána robotická koule Sphero a druhé tlačítko *„O aplikaci“* s textem popisující za jakým účelem byla aplikace vyvinuta.

Obrázek 36 - Tlačítka vyvolávající informační okna

Nejprve jsou vytvořena dvě tlačítka v grafickém prostředí aplikace. Potom je nutné nakonfigurovat akci po kliknutí na tlačítko a to ve funkční části aplikace. Okno je vytvořeno pomocí funkce *„MessageDialog“*, ve kterém je vytvořena proměnná *„dialog1“* a *„dialog2“*. Ke každému dialogu je přiřazen text. Dále je zde přidán parametr, díky němuž je možno dialog zavřít klávesou *„enter“* a parametr samotného zobrazení okna s informační zprávou.

```
private void OSpheru(object sender, RoutedEventArgs e)
{
    MessageDialog dialog1 = new MessageDialog("Sphero je ovladatelná robotická koule,
    dialog1.CancelCommandIndex = 1;
    dialog1.ShowAsync();
}
```

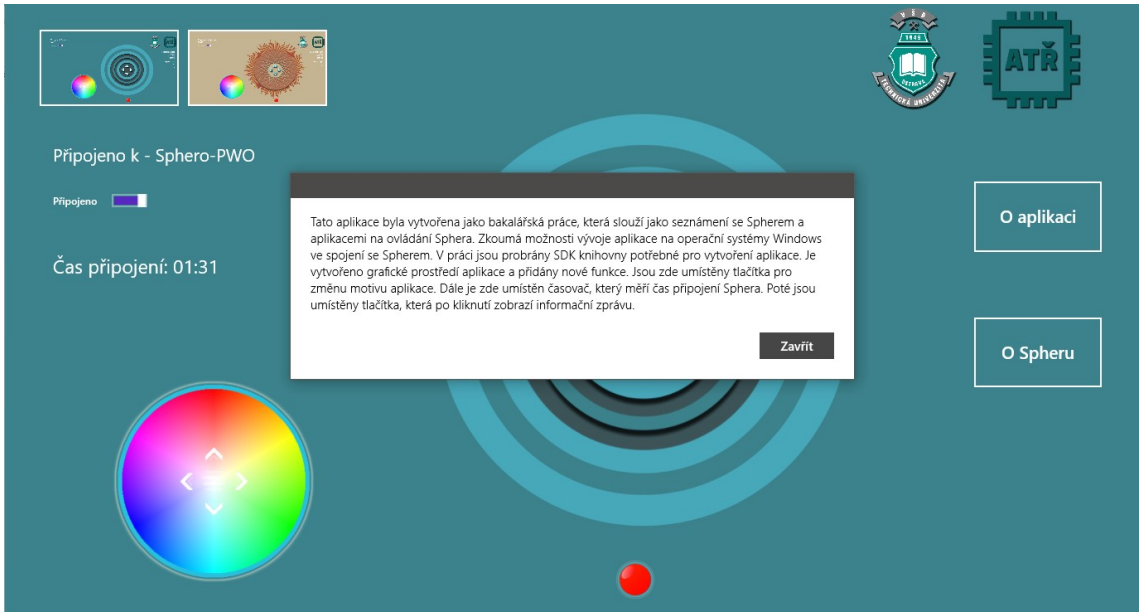
Obrázek 37 - Ukázka kódu - definování tlačítka informačního okna

Tlačítka jsou tak jako u předchozích funkcí vloženy do ohraničení. To v sobě obsahuje parametry pro umístění a výchozí viditelnost. Viditelnost je nastavena na hodnotu – neviditelný, kvůli funkci načítací obrazovka.

```
<Grid x:Name="infoTlacitko" Visibility="Collapsed" HorizontalAlignment="Right"
VerticalAlignment="Top">
<Button Content="O Spheru" HorizontalAlignment="Right" " VerticalAlignment="Top"
Click="OSpheru" Height="107" Width="192"/>
<Button Content="O aplikaci" HorizontalAlignment="Right" VerticalAlignment="Top"
Click="OAplikaci" Height="109" Width="192"/>
</Grid>
```

Obrázek 38 - XAML - tlačítka pro otevření informačních oken

Informační okno se po kliknutí na tlačítko objeví uprostřed obrazovky a je možno jej vypnout kliknutím na tlačítko zavřít, nebo když je aplikace použita v zařízení s klávesnicí je možno okno zavřít klávesou *„enter“*.



Obrázek 39 - Otevřené informační okno v aplikaci

6 Závěr

V této práci jsem se zabýval vývojem aplikace pro ovládání robotické koule Sphero. Ze začátku bylo nutné zjistit, co to Sphero je a jaké nabízí možnosti, ať už jako uživateli, ale hlavně jako vývojáři. To zahrnovalo prozkoumání Sphero koule jako hardwaru, zjišťování dostupných aplikací na různých platformách a hlavně možnostech programování a vývoje vlastní aplikace. Toto umožnuje sama společnost tím, že podporuje programátory a vývojáře uvolněním knihoven (SDK). Bez základních vědomostí programovacího jazyka C# není možno na takovém projektu pracovat, musel jsem si tedy tyto základní vědomosti osvojit. Abych mohl používat dostupné knihovny Sphero koule, potřeboval jsem vývojové prostředí, které mi umožnilo prozkoumat, co tyto knihovny obsahují. Proto jsem využil vývojové prostředí Visual Studio od firmy Microsoft. Aplikace bude určena pro operační systém Windows, proto se toto vývojové prostředí k vytváření nové aplikace více než hodí.

Dalším pokračováním ve vývoji bylo vytvořit pomocí SDK knihoven aplikaci na ovládání Sphera. Nejprve bylo nutné vyzkoušet, jestli se dá koule připojit k aplikaci, zda reaguje na příkazy a prozkoumat celou funkčnost. Poté jsem mohl začít se samotnou tvorbou aplikace. Krom vytváření grafického designu jsem pracoval také na funkční části aplikace jako například nastavení výchozí barvy Sphero koule po připojení nebo změna motivu aplikace pomocí tlačítek. To zahrnovalo seznámení s technologií WPF pro vytváření aplikací s grafickým prostředím a základní nastudování jazyku C#. K vytvoření grafického rozhraní aplikace byl využit značkovací jazyk XAML, který odděluje návrh grafiky od funkční části programu.

Výstupem bakalářské práce je samotná aplikace a návody do cvičení, podle kterých si studenti budou moci osvojit základní principy tvorby aplikace ve spojení knihoven SDK a robotické koule Sphero.

Samotný návrh aplikace byl poměrně bezproblémový. Přenesení myšlenek do vývoje aplikace bylo již obtížnější. Potýkal jsem se s problémy softwaru, také nebylo jednoduché využití již vytvořených objektů a jejich definování. Po vyřešení všech obtíží jsem nakonec projekt dokončil. V práci zůstává prostor pro další řešení, nápady, návrhy a inovace. Což může být námětem diplomové práce.

SEZNAM POUŽITÉ LITERATURY

ALBAHARI, B., DRAYTON, P., MERRILL, B.: C# Essentials., O'Reilly Media, 2002, 216 s., ISBN 0-596-00315-3.

HILYARD, J., TEILHET, S.: C# 3.0 Cookbook., O'Reilly Media, 2007. 886 s., ISBN 0-596-51610-X.

ITUNES. *Itunes*: [online]. 2016 [cit. 2016-04-04]. Dostupné z: https://itunes.apple.com

KAČMÁŘ, D.: Programujeme .NET aplikace ve Visual studiu .NET., Praha: 2001, 335 s. ISBN 80-7226-569-5.

MICROSOFT. Vývoj mobilních řešení pro více platforem. *Visual Studio*: [online]. Praha: Microsoft, 2016 [cit. 2016-01-13]. Dostupné z: https://www.visualstudio.com/features/mobile-app-development-vs

MYSLIVEČEK, D. Sphero 2.0: Robotická koule pro vás i vaše mazlíčky. *Svět androida:* [online]. 2014 [cit. 2015-5-15]. Dostupné z: http://www.svetandroida.cz/sphero-2-0-recenze-201410

PROSISE, J.: Programování v Microsoft .NET., Computer Press, 2003, 736 s., ISBN 80 7226 879 1

SPHERO. *Developer center*: [online] Dostupné z: https://developer.gosphero.com/

SPHERO. *Sphero*: [online], 2015 [cit. 2016-02-04]. Dostupné z: http://www.sphero.com/sphero-sprk

VÁVRA, J. WPF pro začátečníky: 1. díl - Úvod do WPF a vytvoření projektu. *Programujte.com*: [online]. webtea.cz, 2014 [cit. 2016-03-12]. Dostupné z: http://programujte.com/clanek/2014051701-wpf-pro-zacatecniky-1-dil-uvod-do-wpf-a-vytvoreni-projektu/

WIKIPEDIA. Vývojové prostředí. *Wikipedia: the free encyclopedia*: [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-6-10]. Dostupné z: https://cs.wikipedia.org/wiki/V%C3%BDvojov%C3%A9_prost%C5%99ed%C3%AD

.

SEZNAM OBRÁZKŮ